オープンCAE勉強会@富山（第44回）
2016年6月18日
2016年7月4日改訂
2016年7月27日改訂

rhoCentralFoamを用いた衝撃波シミュレーション

- dynamicRefineFvMeshの利用 その1 -

中山 勝之 （オープンCAE勉強会＠富山）

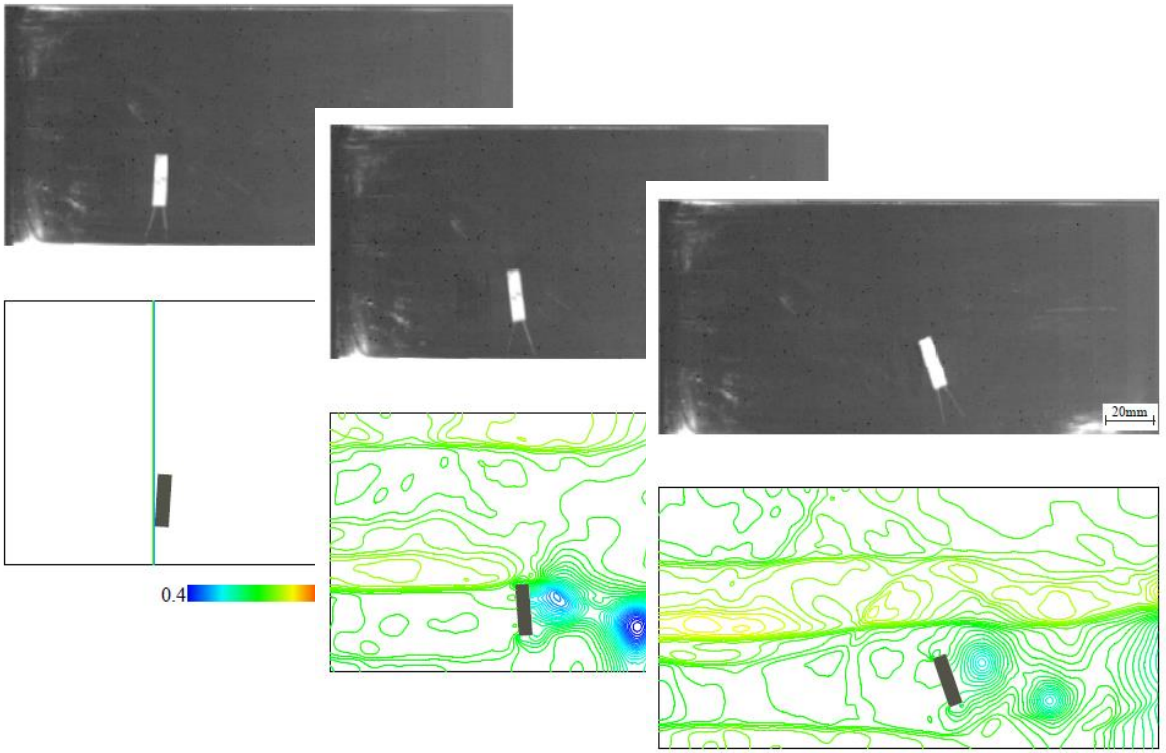
衝撃波を伴う流れの数値シミュレーション

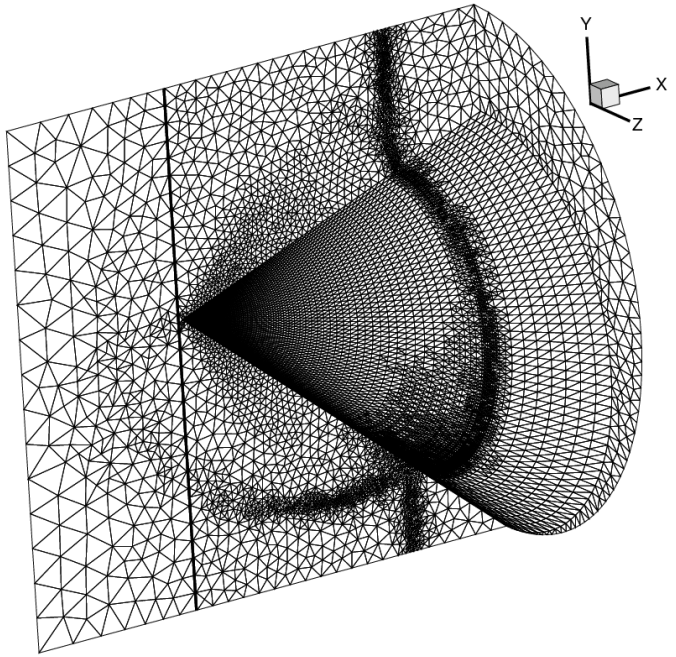
インハウス・コードを使用した解析の経緯

格子系：直交格子（カットセル法）
非構造格子(2次元：三角形メッシュ, 3次元：テトラメッシュ, 解適合格子法)

空間離散化：有限体積法
MUSCL法による高次精度化

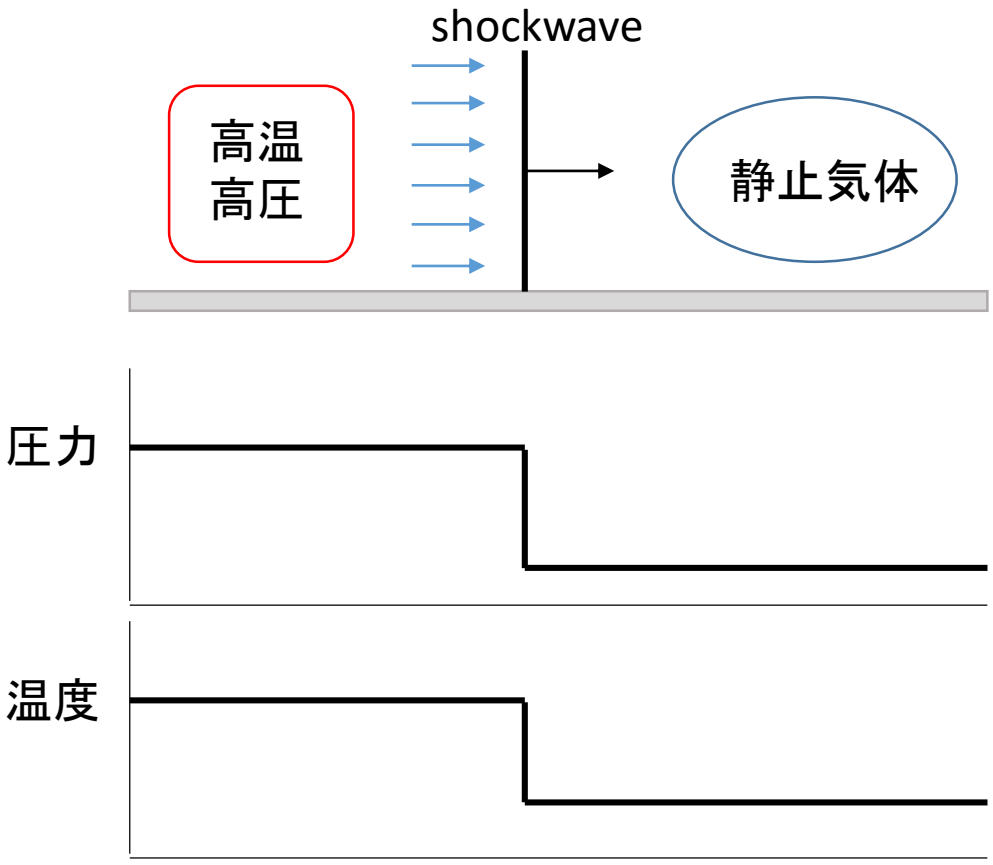
時間積分：2段階Runge-Kutta法（2次精度）





衝撃波とは

衝撃波：音速を超えて伝わる不連続な波

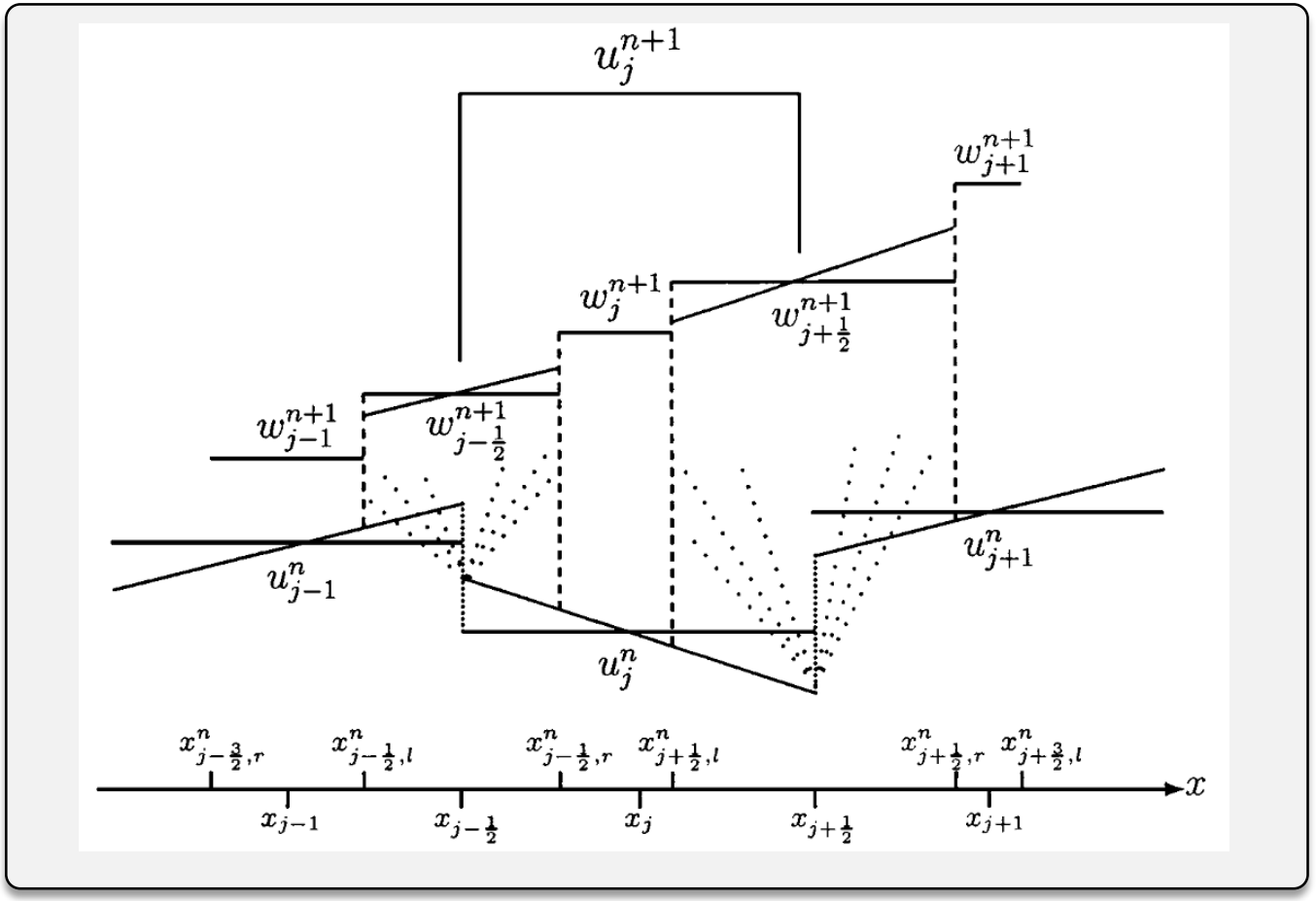


衝撃波が静止気体中を通過すると，気体は圧縮されて温度・圧力が上昇し，
衝撃波によって誘起される流れが発生

OpenFOAMを用いた衝撃波シミュレーション

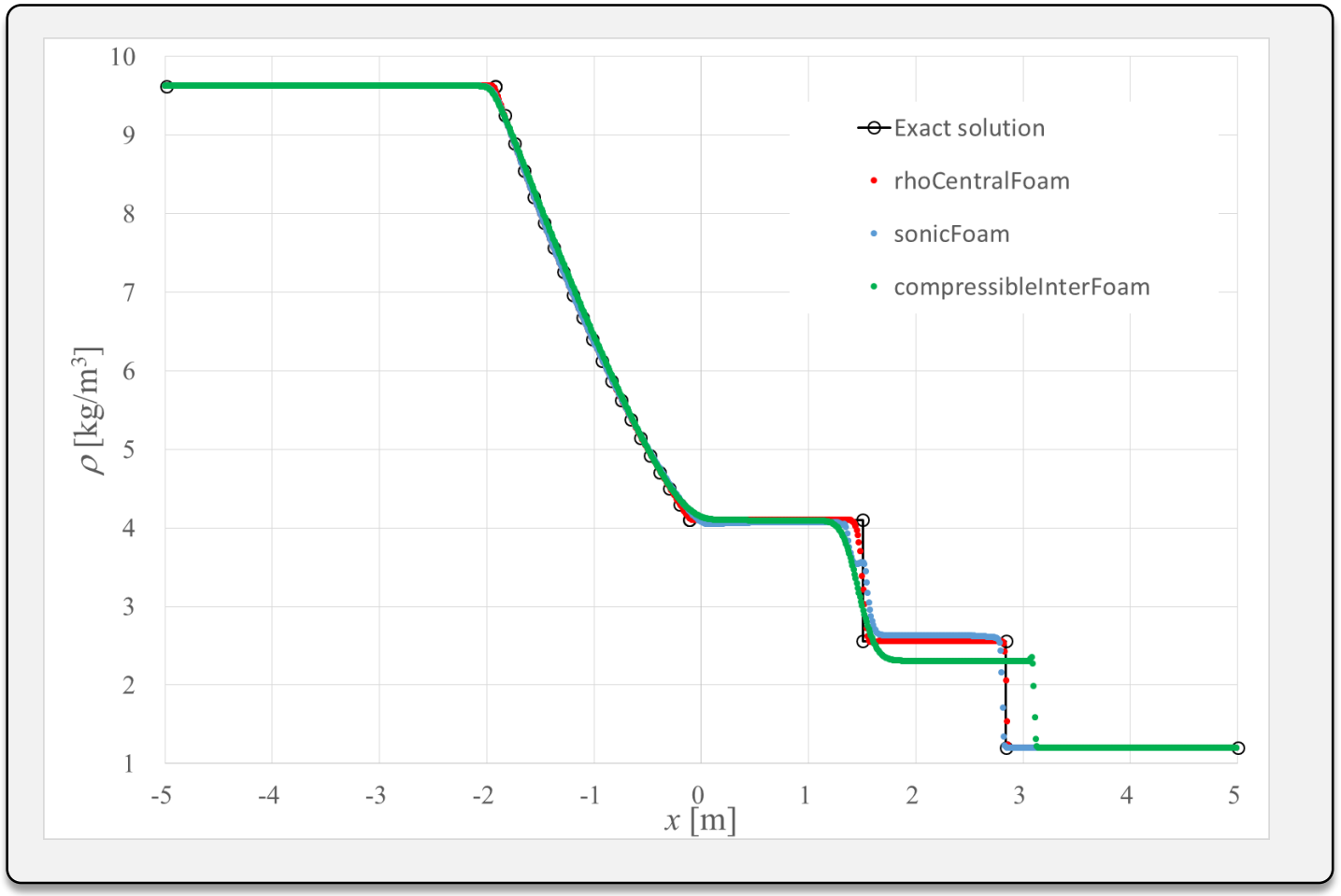
rhoCentralFoamを使用

KuruganovとTadmorにより提案された近似リーマン解法の一つ



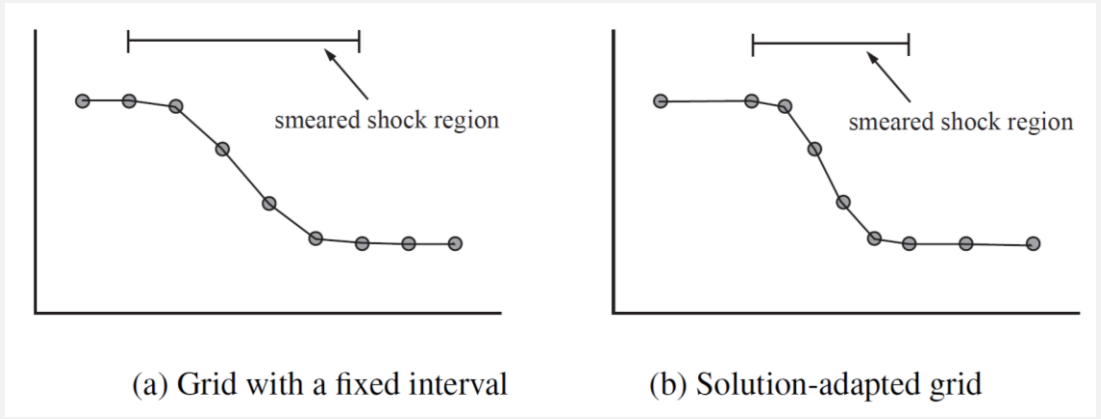
Kurganov, A., Tadmor, E., New High-Resolution Central Schemes for Nonlinear Conservation Laws and Convection-Diffusion Equations, J. Comp. Phys., Vol. 160 (2000), pp. 214 – 282.

厳密解との比較



rhoCentralFoamの計算結果は厳密解に近い結果を示している

衝撃波シミュレーション × 解適合格子法



衝撃波位置に格子を密に配置することにより、みかけ上の衝撃波厚さを小さくすることが可能

一般的に衝撃波の厚さは、気体の平均自由行程の数倍から数十倍程度（数十nmから数百nm）である。衝撃波を捉えるためには、メッシュの高い空間分解能が求められる。したがって衝撃波シミュレーションにおいて、解適合格子法を用いることは、少ない計算メモリで高精度な解析を行う上で有効である。

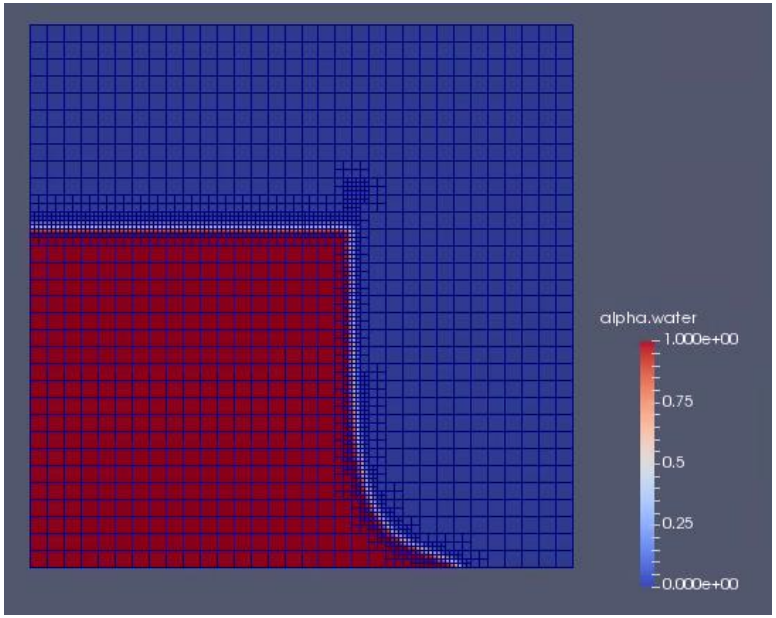
今回の目標

dynamicFvMeshの機能の1つであるdynamicRefineFvMeshを
rhoCentralDyMFoamで使用する

interDyMFoamのチュートリアル

OpenFOAM-3.0.x/tutorials/multiphase/interDyMFoam/ras/damBreakWithObstacle

で使用されている



OpenFOAMの環境について

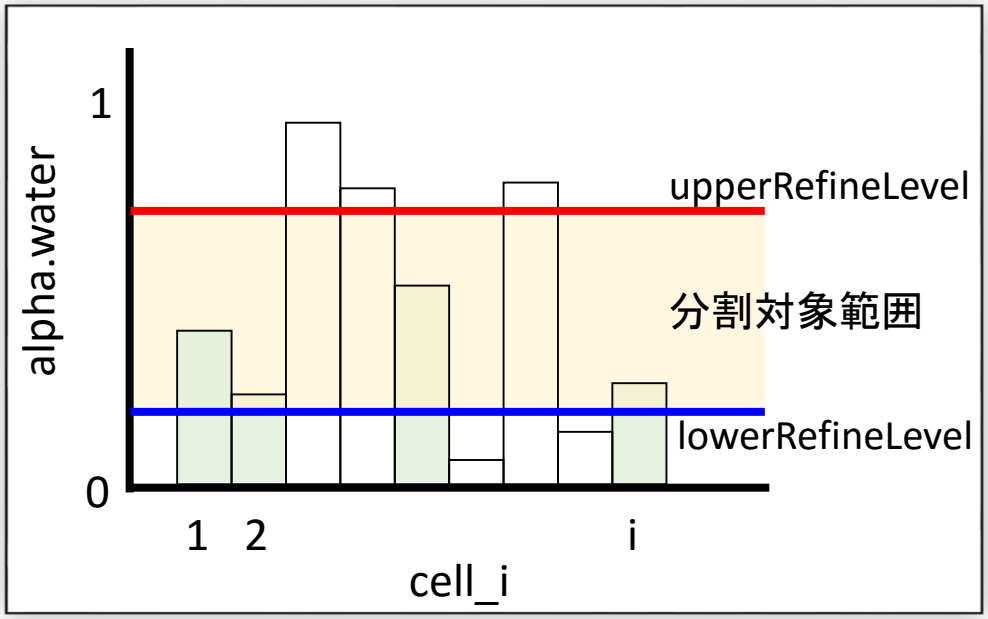
OS : Ubuntu MATE 16.04 LTS (64bit)

OpenFOAM Version : 3.0.x

ケースファイルの確認（1）- interDyMFoam/ras/damBreakWithObstacle-

constant/dynamicMeshDictの一部抜粋

```
dynamicFvMesh   dynamicRefineFvMesh;

dynamicRefineFvMeshCoeffs
{
   // How often to refine
   refineInterval  1;
   // Field to be refinement on
   field         alpha.water;
   // Refine field inbetween lower..upper
   lowerRefineLevel 0.001;
   upperRefineLevel 0.999;
   // If value < unrefineLevel unrefine
   unrefineLevel   10;
   // Have slower than 2:1 refinement
   nBufferLayers   1;
   // Refine cells only up to maxRefinement levels
   maxRefinement   2;
   // Stop refinement if maxCells reached
   maxCells        200000;
   // Flux field and corresponding velocity field. Fluxes on changed
   // faces get recalculated by interpolating the velocity. Use 'none'
   // on surfaceScalarFields that do not need to be reinterpolated.
   correctFluxes
   (
      (phi none)
      (nHatf none)
      (rhoPhi none)
      (ghf none));
   // Write the refinement level as a volScalarField
   dumpLevel       true;
}
```



// If value < unrefineLevel unrefine
unrefineLevel 10;
細分化されたセルが粗雑化の対象となるセルの
フィールド値の基準値を入力
例えば
フィールド値 alpha.waterの最大値は1であるので、
unrefineLevel 10に設定すると、全セルが粗雑化の対象となる
また
unrefineLevel 0に設定すると、一度細分化されたセルは粗雑化
の対象から外れる

ケースファイルの確認（2）- interDyMFoam/ras/damBreakWithObstacle-

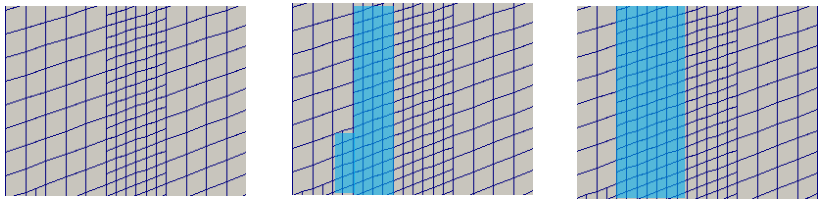
constant/dynamicMeshDictの一部抜粋

```
dynamicFvMesh   dynamicRefineFvMesh;

dynamicRefineFvMeshCoeffs
{
    // How often to refine
    refineInterval  1;
    // Field to be refinement on
    field         alpha.water;
    // Refine field inbetween lower..upper
    lowerRefineLevel 0.001;
    upperRefineLevel 0.999;
    // If value < unrefineLevel unrefine
    unrefineLevel   10;
    // Have slower than 2:1 refinement
    nBufferLayers   1;
    // Refine cells only up to maxRefinement levels
    maxRefinement   2;
    // Stop refinement if maxCells reached
    maxCells        200000;
    // Flux field and corresponding velocity field. Fluxes on changed
    // faces get recalculated by interpolating the velocity. Use 'none'
    // on surfaceScalarFields that do not need to be reinterpolated.
    correctFluxes
    (
        (phi none)
        (nHatf none)
        (rhoPhi none)
        (ghf none));
    // Write the refinement level as a volScalarField
    dumpLevel       true;
}
```

細分化する計算ステップ間隔
1 を設定すると毎ステップ実行
非定常現象の場合小さな値に設定すると良い

細分化セルの上限の制御は
最大分割レベル(maxRefinement)の設定
最大セル数(maxCells)の設定で行う
nBufferLayersは細分化セルのバッファーを設定

ex.
maxRefinement 1
nBufferLayers 0　　4　　8



correctFluxes
再補完するフィールドを設定する
noneを指定することで再補完しない

dumpLevel
細分化レベルを出力
時刻ディレクトリにファイルcellLevelを作成

ソルバーの記述変更

- interDyMFoamを使用したdynamicRefineFvMeshの利用は容易

  フィールド(volScalarField) alpha.water（VOF値）を対象にすることで、
  相の境界面でメッシュを細分化させることが可能

- 衝撃波の位置でメッシュを細分化させるには、密度、圧力変化のフィールド値が必要

- 使用するソルバーに対して細分化させるための基準フィールド値を作成する必要あり

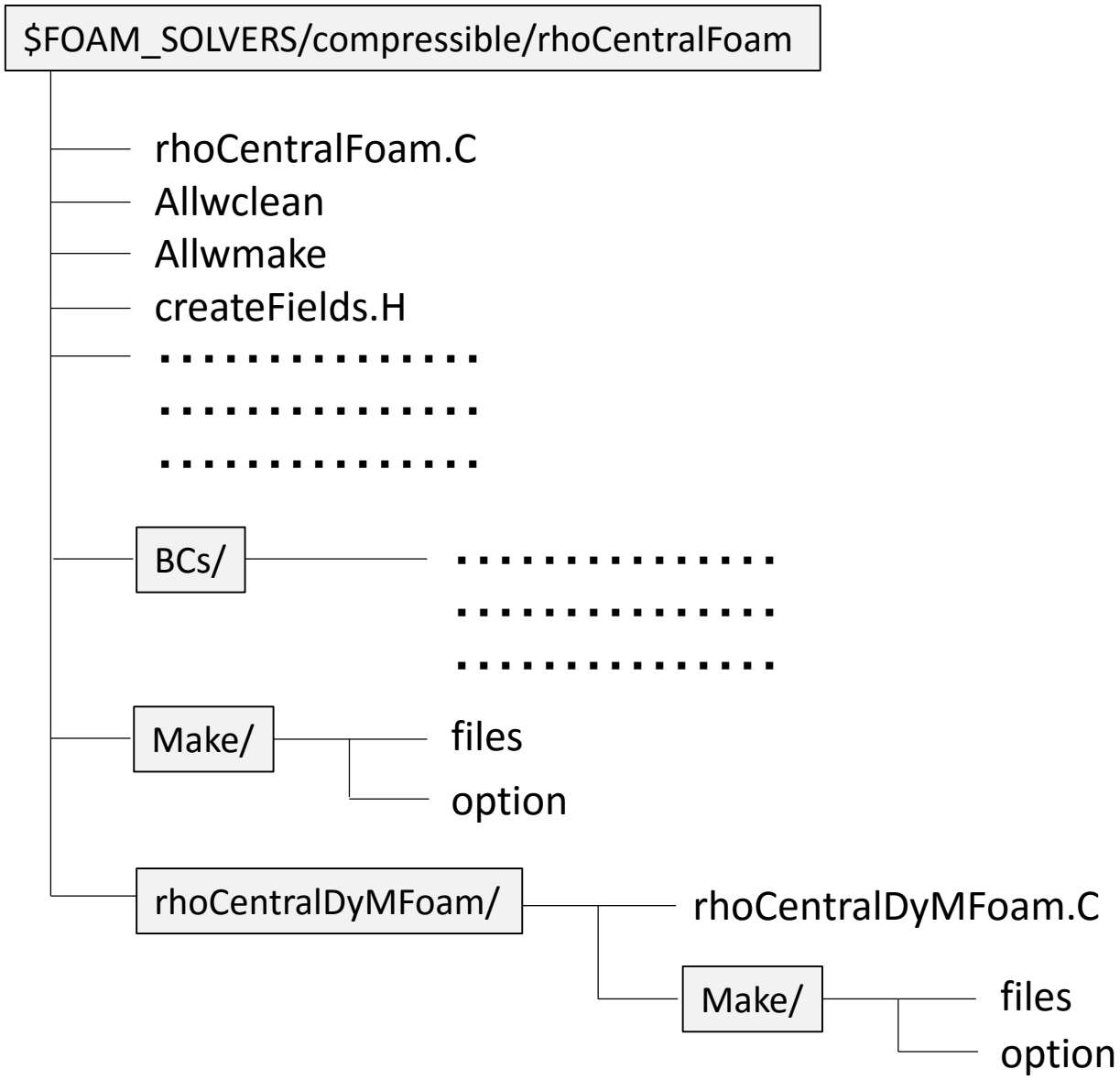
- 一方rhoCentralDyMFoamは存在しているのでdynamicRefineFvMeshは使用可能

以上を踏まえ

作業 1　createFields.Hに基準フィールド値の宣言を記述

作業 2　rhoCentralDyMFoam.Cに基準フィールド値の計算式を記述

ファイル構成(ソルバー, 書き換えた部分のみ抜粋)



前準備（1）

rcFoam, rcDyMFoamという別名のソルバーを作成するための準備

ソルバーのソースファイルをコピーする

```
cp -rf $FOAM_SOLVERS/compressible/rhoCentralFoam  $WM_PROJECT_USER_DIR/rcFoam ⏎
```

ファイル・ディレクトリ名を書き換える

```
cd $WM_PROJECT_USER_DIR/rcFoam ⏎
mv rhoCentralFoam.C rcFoam.C  ⏎
mv rhoCentralDyMFoam rcDyMFoam ⏎
mv rcDyMFoam/rhoCentralDyMFoam.C   rcDyMFoam/rcDyMFoam.C ⏎
```

前準備（2）

赤字の部分を変更

Allwmake

```
#!/bin/sh
cd ${0%/*} || exit 1    # Run from this directory
set -x

(wmake && wmake rcDyMFoam)

# ----------------------------------------------------------------- end-of-file
```

Make/files

```
rcFoam.C

EXE = $(FOAM_USER_APPBIN)/rcFoam
```

Allwclean

```
#!/bin/sh
cd ${0%/*} || exit 1    # Run from this directory
set -x

wclean libso BCs
wclean
wclean rcDyMFoam

# ----------------------------------------------------------------- end-of-file
```

rcDyMFoam/Make/files

```
rcDyMFoam.C

EXE = $(FOAM_USER_APPBIN)/rcDyMFoam
```

作業1　createFields.Hに基準フィールド値の宣言を記述

createFields.H

```
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
volScalarField normalMagGradRho
    (
        IOobject
        (
            “normalMagGradRho”,
             runtime.timeName(),
             mesh,
             IOobject::NO_READ,
             IOobject::AUTO_WRITE
         ),
         mesh,
         dimensionSet(0, 0, 0, 0, 0, 0, 0)
);
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
```

今回作成する基準フィールド値は

正規化された密度勾配（単位は無次元）

作業2　rhoCentralDyMFoam.Cに基準フィールド値の計算式を記述

rcDyMFoam/rcDyMFoam.C

```
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
        turbulence->correct();

//add
        volScalarField magGradRho = mag(fvc::grad(rho));
        normalMagGradRho = magGradRho / max(magGradRho);
//add-end

        runtime.write();
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
```

normalMagGradRhoの計算式を記述する

$$\text{normalMagGradRho} = \frac{|\nabla \rho_{iCell}|}{\max(|\nabla \rho_{iCell}|)}$$

ソルバーのコンパイル

```
cd $WM_PROJECT_USER_DIR/rcFoam ⏎
./Allwmake ⏎
```

$WM_PROJECT_USER_DIR/platforms/$WM_OPTIONS/binに
rcFoamとrcDyMFoamが作成される

ケースファイルの作成

- $FOAM_TUTORIALS/compressible/rhoCentralFoam/shockTubeをベースに作成

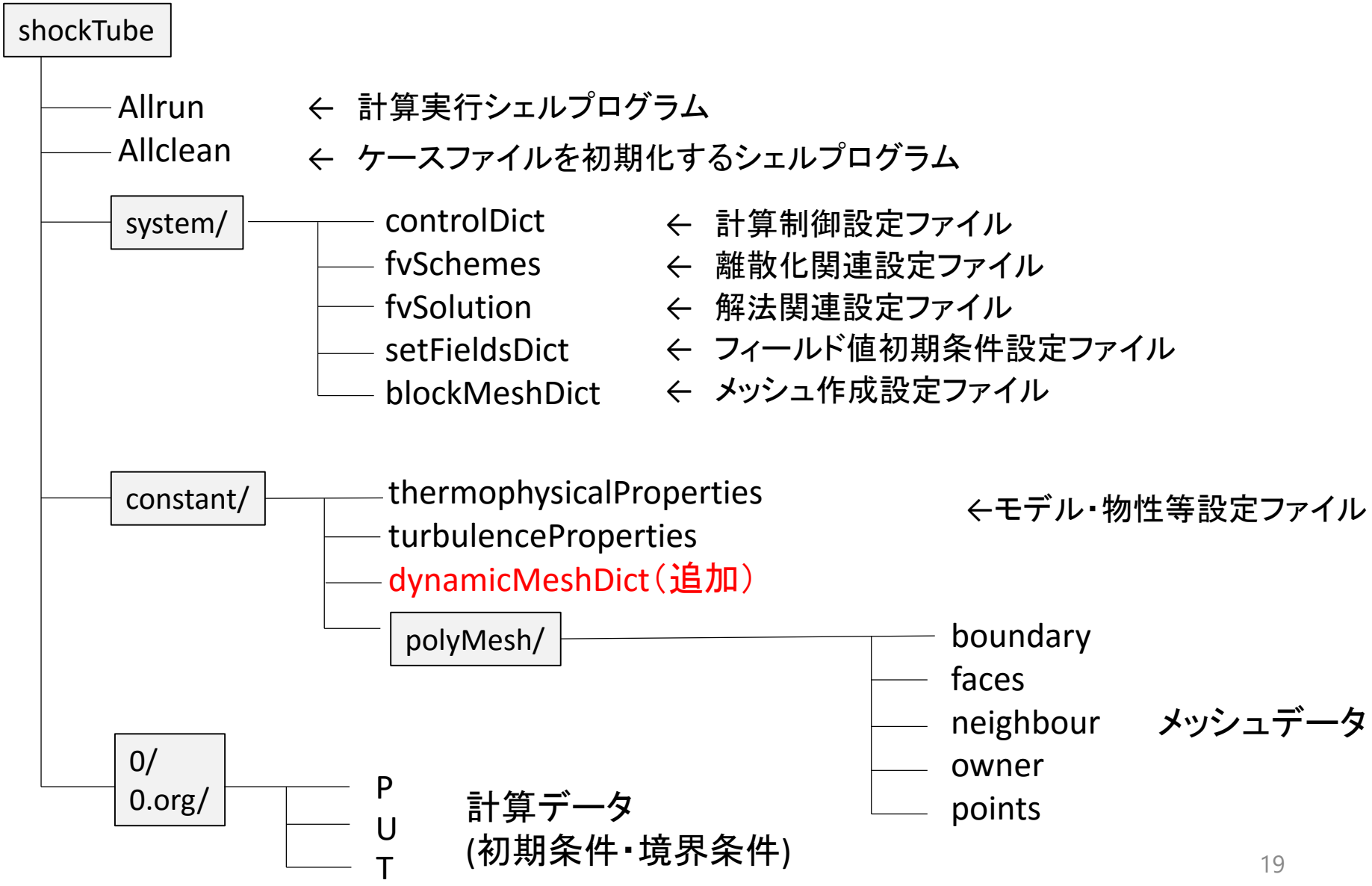
ケースファイルをコピーする

```
cp -rf $FOAM_TUTORIALS/compressible/rhoCentralFoam/shockTube $WM_PROJECT_USER_DIR/ ⏎
```

作業 1　境界タイプが empty だと計算できないので
境界面emptyの境界タイプを変更
(blockMeshDict, 0.org/p,U,Tの書き換え)

作業 2　dynamicMeshDictディクショナリを作成

ファイル構成(ケースファイル)



ケースファイルの修正（1）

赤字の部分を変更

system/controlDict

```
.....

application    rcDyMFoam

.....
```

境界条件はすべての境界面でslipを選択

0.org/p

```
.....
boundaryField
{
   sides
   {
      type         slip;
   }

   empty
   {
      type         slip;
   }
}
.....
```

0.org/U

```
.....
boundaryField
{
   sides
   {
      type         slip;
   }

   empty
   {
      type         slip;
   }
}
.....
```

0.org/T

```
.....
boundaryField
{
   sides
   {
      type         slip;
   }

   empty
   {
      type         slip;
   }
}
.....
```

ケースファイルの修正（2）

system/blockMeshDict

```
.....................................

boundary
(
    sides
    {
        type patch;
        faces
        (
            (1 2 6 5)
            (0 4 7 3)
        );
    }
    empty
    {
        type patch;
        faces
        (
            (0 1 5 4)
            (5 6 7 4)
            (3 7 6 2)
            (0 3 2 1)
        );
    }
);
.....................................
```

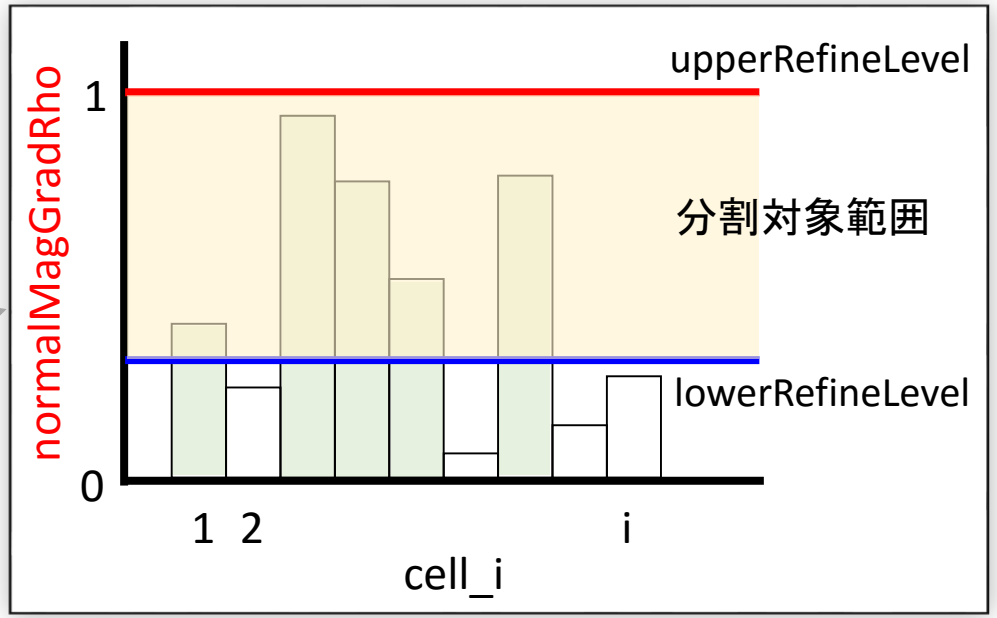
emptyの境界タイプをpatchに変更

dynamicMeshDictの内容

constant/dynamicMeshDict

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      dynamicMeshDict;
}
dynamicFvMesh   dynamicRefineFvMesh;

dynamicRefineFvMeshCoeffs
{
    // How often to refine
    refineInterval  1;
    // Field to be refinement on
    field           normalMagGradRho;
    // Refine field inbetween lower..upper
    lowerRefineLevel 0.2;
    upperRefineLevel 1.0;
    // If value < unrefineLevel unrefine
    unrefineLevel   10;
    // Have slower than 2:1 refinement
    nBufferLayers   2;
    // Refine cells only up to maxRefinement levels
    maxRefinement   3;
    // Stop refinement if maxCells reached
    maxCells        200000;
    // Flux field and corresponding velocity field. Fluxes on changed
    // faces get recalculated by interpolating the velocity. Use 'none'
    // on surfaceScalarFields that do not need to be reinterpolated.
    correctFluxes
    (
        (phi none)
        (pos none)
        (neg none)
);
    // Write the refinement level as a volScalarField
    dumpLevel       true;
}
```

$$\text{normalMagGradRho} = \frac{|\nabla \rho_{iCell}|}{\max(|\nabla \rho_{iCell}|)}$$



// If value < unrefineLevel unrefine
unrefineLevel 10;
細分化されたセルが粗雑化の対象となるセルの
フィールド値の基準値を入力
例えば
フィールド値 normalMagGradRhoの最大値は1であるので、
unrefineLevel 10に設定すると、全セルが粗雑化の対象となる
また
unrefineLevel 0に設定すると、一度細分化されたセルは粗雑化
の対象から外れる

ケースファイルの実行

```
cd $WM_PROJECT_USER_DIR/shockTube ⏎
./Allrun ⏎
```

Allrun

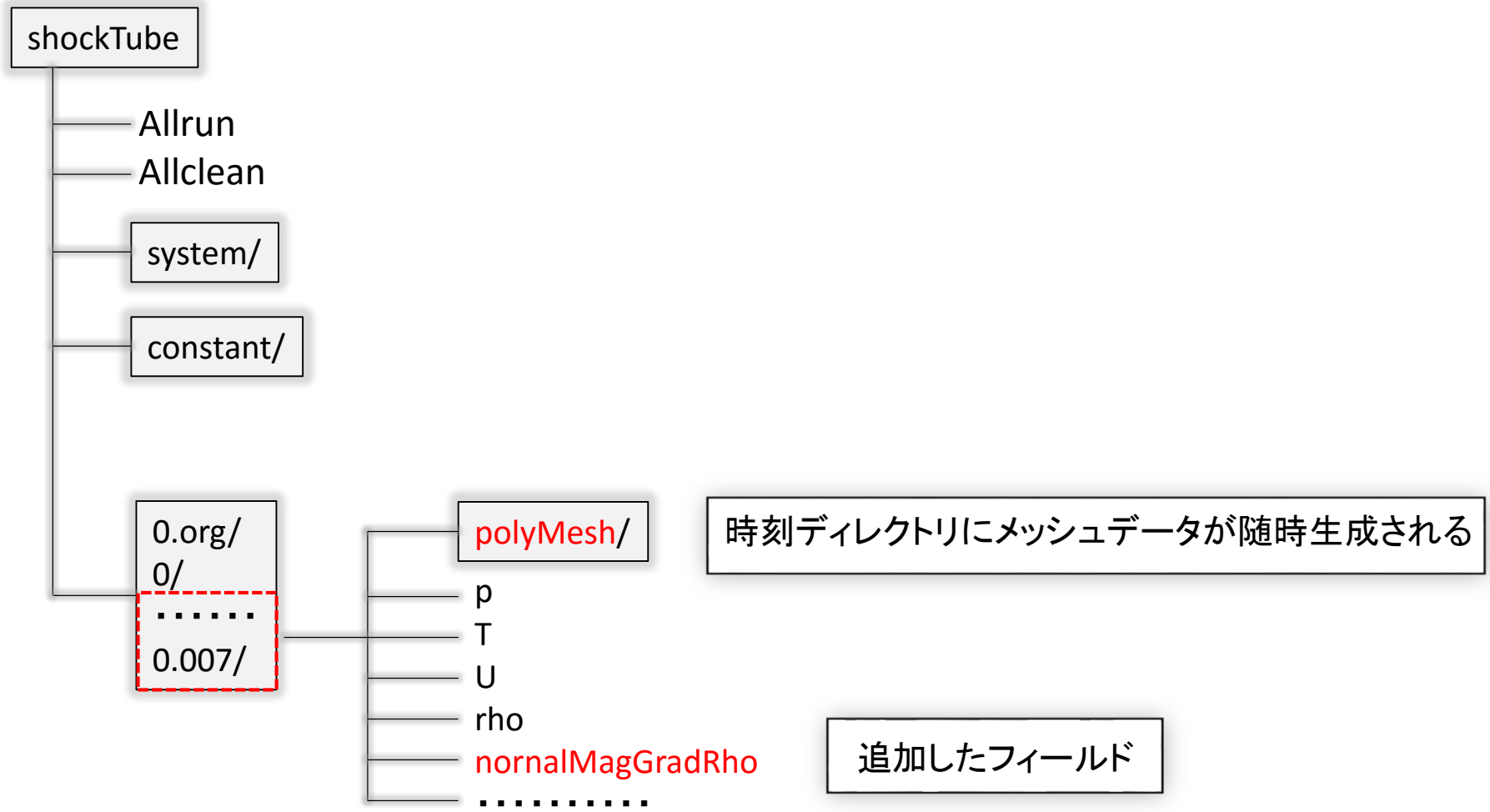
```
#!/bin/sh
cd ${0%/*} || exit 1    # Run from this directory

# Source tutorial run functions
. $WM_PROJECT_DIR/bin/tools/RunFunctions

runApplication blockMesh
runApplication setFields
runApplication `getApplication`
```

コマンドrunApplication
変数　getApplication
を使用するためにRunFunctionsの設定を読み込む

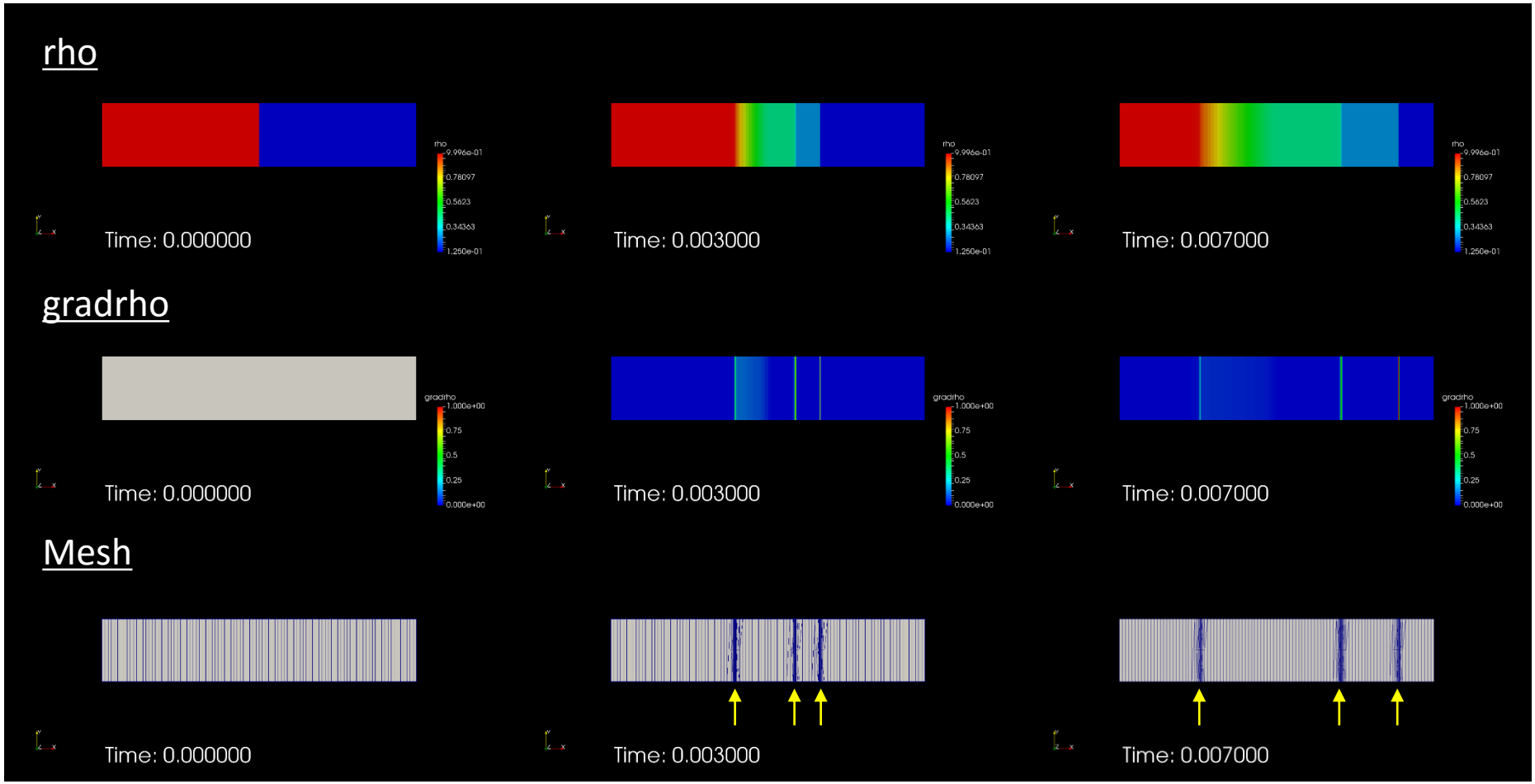
runApplication command ⏎は
command > log.command 2>&1 ⏎
を意味する
log.commandが既に存在する場合はコマンドが実行されないので、予め削除する

getApplicationは
system/controlDictに書かれている
applicationの変数（ソルバー名）を取得する

計算実行後のファイル構成 (ケースファイル)



計算結果



衝撃波、接触面、膨張波波頭でメッシュが細分化されていることを確認

まとめ

dynamicRefineFvMeshをrhoCentralDyMFoamで使用するために（ほんの少し）ソースの改造を行った

次回：別の方法を紹介（その2に続く）

参考文献

以下のHPを参考にさせていただきました。ありがとうございます。

PENGUINTIS:アダプティブメッシュ

http://www.atmarkit.co.jp/ait/articles/0111/29/news003.html